New Framework For Improving Big Data Analysis Using Mobile Agent

Youssef M. ESSA
Software Engineering Department,
Etisalat Corporation,
Cairo, Egypt

Gamal ATTIYA
Computer Science & Engineering
Dept., Faculty of Electronic
Engineering, Menoufia Uni.,
Menouf 32952, Egypt

Ayman EL-SAYED,
IEEE Senior Member
Computer Science & Engineering
Dept., Faculty of Electronic
Engineering, Menoufia Uni.,
Menouf 32952, Egypt

***Abstract*—the rising number of applications serving millions of users and dealing with terabytes of data need to a faster processing paradigms. Recently, there is growing enthusiasm for the notion of big data analysis. Big data analysis becomes a very important aspect for growth productivity, reliability and quality of services (QoS). Processing of big data using a powerful machine is not efficient solution. So, companies focused on using Hadoop software for big data analysis. This is because Hadoop designed to support parallel and distributed data processing. Hadoop provides a distributed file processing system that stores and processes a large scale of data. It enables a fault tolerant by replicating data on three or more machines to avoid data loss. Hadoop is based on client server model and used single master machine called NameNode. However, Hadoop has several drawbacks affecting on its performance and reliability against big data analysis. In this paper, a new framework is proposed to improve big data analysis and overcome specified drawbacks of Hadoop. These drawbacks are replication tasks, Centralized node and nodes failure. The proposed framework is called MapReduce Agent Mobility (MRAM). MRAM is developed by using mobile agent and MapReduce paradigm under Java Agent Development Framework (JADE).**

Keywords—Mobile Agent; JADE; Big Data Analysis; HDFS; Fault Tolerance

I. INTRODUCTION

The collection of data sets are so large and complex that become difficult to process using on-hand database management tools or traditional data processing applications referred to "Big Data". The value of big data to an organization falls into two categories: analytical use and enabling new products. Extracting information and something intelligence from these big data sets, commonly referred to as big data analytics. Big data analytics can reveal insights hidden previously by data too costly to process such as peer influence among customers, revealed by analyzing shoppers' transactions and social and geographical data. Big data analytics is shown to be useful in several scenarios; analytics enable web data mining and enable extracting business intelligence. The primary goal of big data analytics is to help companies to make better business decisions. But, analysis of large data sets in real-time requires a framework like MapReduce to distribute the work among tens, hundreds or even thousands of computers. So, many companies focused on using Hadoop for big data analysis.

Hadoop is an open source software framework written in Java by Doug cutting and Michael Cafarella [1]. Hadoop enables distributed, data intensive and parallel applications by dividing big data into smaller data blocks. These data blocks are divided into smaller partitions such that each data block processes a different partition in parallel [2]. By using Hadoop, there is no limit of storing and processing data by computational technique called MapReduce [3] and in [4-5], the authors proposed a design of an adaptive scheme to efficiently manage the power peaks for MapReduce clusters. Hadoop provides a distributed file processing system that stores and processes a large scale of data [6]. It enables a fault tolerant by replicating data on three or more machines to avoid data loss [7-8], but this method causes some problems. The first problem is about increasing the amount of data that executes on machine by replicating each block of data in two or more machines. The second one, the full system is down when the master machine failed.

So, in this paper presents a new strategy called MapReduce Agent Mobility (MRAM) to improve big data analysis and overcome the drawbacks of Hadoop. The proposed framework is developed by using mobile agent and MapReduce paradigm under Java Agent Development Framework (JADE).JADE is a promising middleware based on the agent paradigm because it supports generic services such as communication support, resource discovery, content delivery, data encoding and agents mobility [9,10].

Indeed, there are seven reasons for using mobile agents as follows:

1) *Reduce the network load,*
2) *Overcome network latency,*
3) *Encapsulate protocols,*
4) *Execute asynchronously and autonomously,*
5) *Adapt dynamically,*
6) *Naturally heterogeneous and robust, and*
7) *Fault-tolerant [11].*

So, the mobile agent is used with Hadoop to overcome the problems faced Hadoop. In the proposed strategy, mobile agents send both code and data to any machine. The machine can react dynamically for any changes in the environment. Furthermore, if a machine or environment down, the mobile agent can migrate to another machine with code and data.

The rest of this paper is organized as follows: Section II describes Hadoop architecture, workflow, and drawbacks. Section III presents the basic concepts of JADE and Mobile Agent. Section IV introduces the proposed framework namely MapReduce Agent Mobility (MRAM). Section V presents a comparative study and performance evaluation of the proposed strategy and Hadoop. Finally, the paper is concluded in Section VI.

II. HADOOP ARCHITECTURE AND WORKFLOW

This section presents both the architecture of Hadoop and its workflow for big data analysis as follow:

A. Hadoop Architecture

Hadoop architecture consists of a Hadoop Distributed File System (HDFS) and a programming framework MapReduce. HDFS stores big files across machines in a large cluster. Each file is stored as a sequence of blocks. Each block is sent to three or more machines for fault tolerance. Hadoop uses MapReduce method for processing data allocated on each node [12, 13, and 14].

1) HDFS

HDFS is a very large distributed file system [15, 16] that is available hardware and provides fault tolerance as well as have high throughput. Many big companies believe that within a few years, more than a half of the world's data will be stored in Hadoop. HDFS stores files as a series of blocks and replicates the data blocks for fault tolerance. HDFS is designed to store big data set, and provides global access to files in the cluster. HDFS stores metadata on a dedicated server, called "NameNode". Application data is stored on other servers called "DataNodes". All servers are fully connected and communicate with each other using TCP-based protocol [2, 15]. HDFS architecture is broadly divided into following four parts as the follows: NameNode, DataNode, JobTracker to determine the location of data and Task Tracker overseeing overall Map Reduce job execution.

a) NameNode

The NameNode is responsible of managing all metadata and file system actions. It handles the file system namespace operations like open, close, and renames both file and directory. Also, it makes all decisions regarding replication of blocks. NameNode maintains the tree of namespace and maps the file blocks to DataNodes (i.e. the physical location of file's data). A single NameNode is considered a bottleneck for handling requests in scientific application environments [12, 17].

b) DataNode

The DataNode stores data in the Hadoop file system, Each DataNode stores data blocks on behalf of local or remote clients. Each block is saved as a separated file in the node's local file system. On startup, DataNode connects to the NameNode and performs a handshake. The purpose of the handshake is to verify the name space IDand the software version of DataNode. If NameNode does not match DataNode, the DataNode automatically shuts down. After the handshake is successful, the DataNode registers with the NameNode. DataNodes persistently store their unique storage IDs. The storage ID is an internal identifier of the DataNode which makes it as recognizable even if it is restarted with a different IP address or port. The storage ID is assigned to the DataNode, when it registers with the NameNodeon the first time and never changes later.The DataNode then responds to the requests that coming from the NameNode, for the file system operations. The DataNodes service the read, writing and file replication requests based on the direction from which NameNode coming [8, 18].

c) JobTracker

The JobTracker talks to the NameNode to determine the location of the data. JobTracker schedules individual maps reduces or intermediate merging operations to specific machines. It monitors the success and failures of these individual tasks. Also, it works to complete the entire batch job. If a task fails, the JobTracker will automatically re-launch the task, possibly on a different node, up to a predefined limit of retries [17, 18].

d) TaskTracker

The JobTracker is the master overseeing the overall execution of a MapReduce job. The TaskTrackers manage the execution of individual tasks on each slave node. Although, there is a single Task Tracker per slave node, each Task Tracker can spawn multiple Java Virtual Machines (JVMs) to handle many maps or reduces the tasks in parallel. The TaskTrackers also transmit heartbeat messages to the JobTracker, usually every a few minutes, to reassure the JobTracker that is still alive [11, 17].

2) MapReduce

In MapReduce [13], the first step is the map job which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job then takes the output from a map as input and combines those data tuples into a smaller set of pairs. The map function can run independently on each key/value pair, exposing enormous amounts of parallelism. Similarly, the reduce function can run independently on each intermediate key, exposing significant parallelism as well. Similar to other distributed systems, MapReduce also constitutes a master and a set of workers.The master is called JobTracker, while the workers are called TaskTrackers [14, 15].

B. Hadoop Workflow

The workflow of Hadoop is shown in Fig. 1. It has the following steps:

1) Input text files to a platform.

2) Server portioning file to blocks with the same size, then assigns a block of data to each computing node.

3) The compute node runs map on the input data and producing intermediate data pair for every word, then sends its intermediate data pairs to the node designated to perform the reduce operation.

4) The reduce operation counts the number of occurrences of each word using the values and emits it as a key-value pair.

5) Server receives the results and outputs the list.

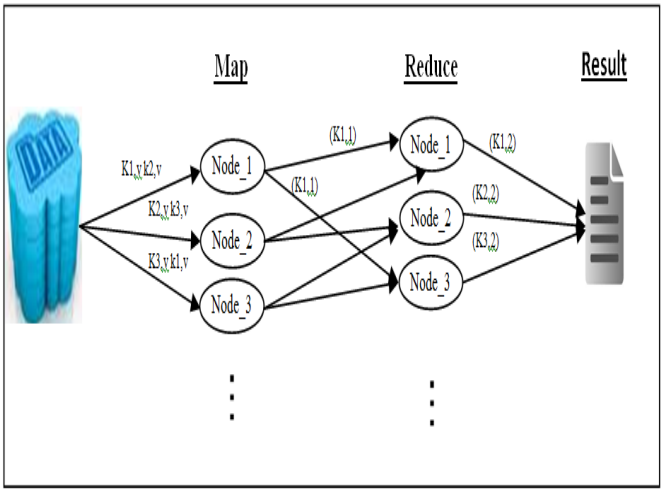

Fig. 1. Workflow for Hadoop.

C. Hadoop Drawbacks

From the architecture of Hadoop and its workflow of data computation, there are many drawbacks of Hadoop. These drawbacks are:

1) Hadoop needs high memory and big storage to apply replication technique.

2) Hadoop supports allocation of tasks only and do not have strategy to support scheduling of tasks.

3) Still single master (NameNode) which requires care

4) Load time is long.

These drawbacks effect on both the performance and reliability of Hadoop against big data analysis. Therefore, it is necessary to develop a new framework or modify some Hadoop features to overcome Hadoop limitations and improve its performance and reliability. So, in this paper, a new framework is proposed to overcome the drawbacks of Hadoop and improve big data analysis.

III. Basic Concepts of Jade and Mobile Agent

A. JADE Architectural Model

In the recent years, there are many platforms that can support agent mobility and developing distributed application. JADE is a promising middleware based on the agent paradigm. It supports generic services such as communication support, resource discovery, content delivery, data encoding and so on [9,10].The architectural of JADE contains both the libraries required to develop application agents and the run-time environment that provides the basic services. These services include agent identification and agent communication. The instance of JADE is called "*Container*" and the set of all containers is called platform [10].

B. Mobile Agent

A mobile agent (MA) is a software abstraction that can migrate during execution across a heterogeneous or homogeneous network. It has the ability to suspend its execution according to some factors and resume it in another machine.

Characteristics of MA: There are several characteristics can be defined the structure of the MA [19]:

State: the main characteristic of the MA. It can stop execution on one machine and resume execution on another machine. The state depends on two factors:

1) Execution state, which is a runtime state including its program counter and stack.

2) Object state, which stores the current values of its variables.

Implementation: it is the program code that defines the tasks behavior. If java is used as MA platform, classes present the implementation code. In this manner, there are two ways to make the required classes available to the MA:

1) Taking the entire required classes during its itinerary and uses it any time anywhere.

2) Taking some of the required classes and once the MA need a class that is not available, it retrieves it from remote location. This operation called Code-On-Demand technique, and it is a common technique in distributed network systems.

Interface: MA collaborates with other agents to handle the assign job. The Interface is required to make the communication possible between agents.

Unique Identifier: it is a unique ID define agent during its lifetime. It used as a key that needed to refer for a specific agent especially, when it travels all over the network.

Itinerary: it is the group of addresses created once the MA life starts that defines the agent journey around the network.

Principals: it is the information of individual, organization or corporation that MA belongs to. Principles are needed to authenticate the MA who dispatched to several destinations on the network.

Advantages of mobile agent: There are many advantages for using mobile agent to solve many problems on distributed application [9, 10].

Reduce network traffic: the cooperation in a distributed system is often achieved using communication protocols. These protocols transfer a large volumes of data stored at remote hosts over the network to a central processing site resulting in high network traffic. At this case, mobile agent uses alternative communication protocols.

Off-line tasks: network connections may be fail at any time. Agents can solved this problem by perform off-line tasks and send results to server application when it come back online.

Support for heterogeneous environments: MA can work on top of any operating system with the same its mobility framework.

Fault tolerance: Mobile agents react dynamically and autonomously to the changes in their environment. If

a host is being shut down or platform is down, all agents executing on that machine will be warned and given time to dispatch themselves and continue their operation on another host in the network [20, 21].

Protocol Encapsulation: Protocol encapsulation allows the components of distributed system to communicate and coordinate their activities. MAs provide a solution to the problem of upgrading the protocol code at all locations in the distributed system.

IV. Our Proposed Framework

Our proposed framework is called *MapReduce Agent* Mobility (MRAM). It combines of advantages of mobile agent and MapReduce technique. MRAM framework improves big data analysis and overcomes the drawbacks of Hadoop during three steps.

A. First Step

Hadoop reduces CPU utilization by providing faults tolerance via replication data. The MRAM provides a fault tolerance when machine is failed by reacting dynamically to system change and can move new agent to another machine with code, data and status to continue executing task. In Hadoop, each data block is sent to three or more nodes. New strategy is completely different.

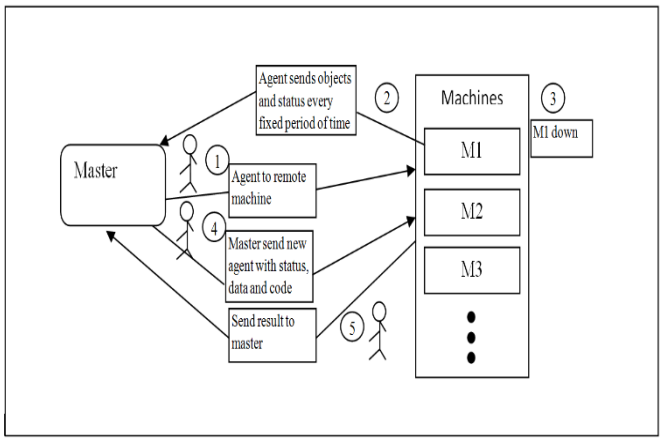

Fig. 2. Steps to solve machine fail problem.

Fig. 2 illustrates the steps to resolve this problem. The main idea of this strategy is each machine in the framework sending a copy of data and status to master machine every fixed time period. In falling machine case, another copy of agent is moving from master machine to a new machine. New agent is carrying copy of code, data and status to completion task.

B. Second Step

The goal of second step is comparing performance of Hadoop and MRAM. The idea of comparative study is applying the same application on Hadoop and MRAM.

The workflow of the proposed MRAM framework is shown in Fig. 3. It has the following steps:

1) Input text files to the platform.

2) Server portioning the file to blocks with the same size

3) An application server assigns a data block to each computing node, but in our approach the server take a task as the other nodes.

4) The computing node runs map on the input data and producing intermediate data pair for every word. It then sends its intermediate data pairs to application server directly to perform the reduce operation.

5) The reduce operation counts the number of occurrences of each word using the values and emits it as a key-value pair and save the result in file or in consol.

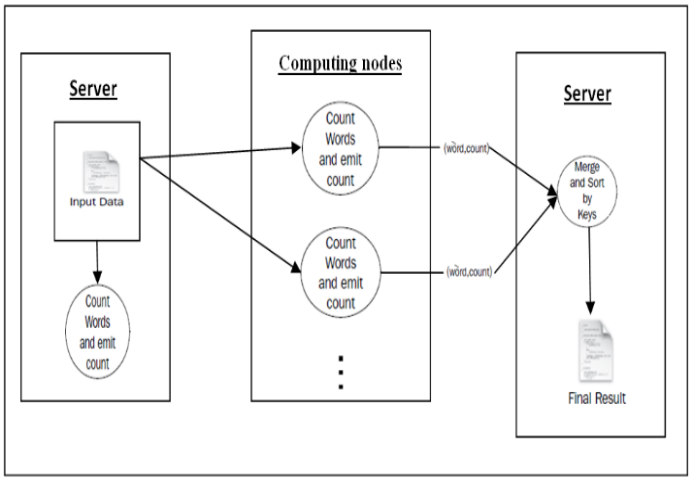

Fig. 3. MRAM Workflow.

C. Third Step

This step uses features of mobile agent. Mobile agent can react dynamically and autonomously to change in their environment. Hadoop is still depending on single node that runs all the services needed to MapReduce task distribution and tracking. The all system is down when a single mode is failed or down. The solution of this problem, the master node is selected when a platform starts working. After that, the master node build linked list involves meta-data. These meta-data contains all information about tasks, dependences among them and information about all machines.

Also, the master machine sends meta-data to all machines through network connection. Subsequently, if any node receives a job, this node is elected as a new master. When the master machine is shutdown or platform is down, all agents executing on master machine will be moved to another host that having a highest IP-address when meta-data is published. The agents continue executing tasks on new machine because it carrying its code, status and data.

A new machine becomes as a master node that is responsible for all acts of server expects to receive result from machines such as Task Tracker and informs all machines about a machine failed. After all agents finished executing tasks, it is waiting to send the result to general server when it comes back online as shown in Fig. 4.

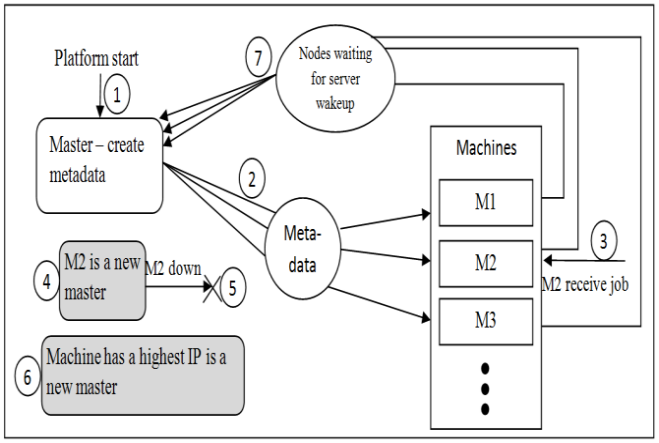

Fig. 4. Steps to solve centralized node problem.

D. Advantages of MRAM

The MRAM framework has several advantages derived from the features of mobile agent and MapReduce technique. The advantages of MRAM are:

1) Support allocation and scheduling tasks.

2) Provides fault tolerance and don't need high memory or big disk to support it.

3) Load time for MRAM is less than that of Hadoop.

4) Solve single master (centralized node) problem by using features of mobile agent.

5) Improve execution time because of no need to huge processing to replication data.

V. Comparative Study and Performance Analysis

It is noted that MRAM improves reliability of Hadoop using mobile agent and investigate the performance of Hadoop and MRAM. The idea of comparative study is applying the same application in the same environment on Hadoop and MRAM.

A. Implementation Environment

In this paper, the Measurements have been carried out by using the following hardware and software components. The specifications of the used hardware and software are shown in Table I, and as follows:

1) Hardware Components: Hardware contains one server namely "Server" and three nodes namely "PC1", "PC2" and "PC3" connected via a LAN.

2) Software Components: Hadoop and MRAM are the main software components. The word count application or multiply two arrays application is applying on each platform.

B. Durability of platforms

As mentioned before, there are three weaknesses for Hadoop. The first weakness, Hadoop is still single master which require care. The second one, Hadoop reduces the CPU utilization by providing faults tolerance via replication data.

TABLE I. Software and Hardware Equipments

	Server	PC1	PC2	PC3
Model	**IBM x3650**	**IBM**		
CPU	Intel Dual Core2Quad 2.56 GHZ	Intel Dual Core2Due 2.53 GHz		
RAM	16GB	2 GB		
Hadoop version	0.20	0.20		
OS	Linux	Linux		
Sun JRE	JRE 7u25	JRE 7u25		
JADE	4.1	4.1		

TABLE II. Data About States of Task.

Time interval in second	0.25	0.5	1	2	4	8
1st snapshot of objects	i=1 j=1988	i=1 j=1983	i=1 j=2088	i=1 j=2001	i=1 j=2001	i=1 j=1912
Last snapshot	i=3 j=2240	i=5 j=1255	i=8 j=338	i=17 j=150	i=38 j=613	i=65 j=2236
Last status sent	i=1 j=1886	i=2 j=1246	i=4 j=1676	i=8 j=2420	i=19 j=1271	i=37 j=871

Sure, these factors effect on the reliability of Hadoop. So the solution is via mobile agent as in MRAM and will clarify each solution separately as follow.

1) Faults Tolerance Techniques

Table II contains the data used to measure the appropriate times periods that sends the in all of them the data and status about tasks. The Framework is using two arrays each one consist of two-dimensional. Theses matrix is multiplying in this step and measuring execution time in different states. First dimension is defined by i and second dimension by j.

The status and data takes in each time period for each array. So, the first measurement value takes after the program began, and then used a fixed period of time to take the status and data of the task. We assume the worst case that occurs if a machine is down just before taking a status and data. All values in table on the basis of this case.

From the Table II, the relationship between time and the amount of data processing is a proportional relationship. Also, the relationship between time and amounts of processes losses is proportional relationship.

Fig. 5 illustrates size of sent data in every period of time, where the amount of data sent is decreased when the period of time increased. But this in turn affects the reliability of the system because when the time period increased will be the possibility of processes losses are larger as shown in Fig. 6. Also, the time needed to send data is greater when the amount of data sent is increased.

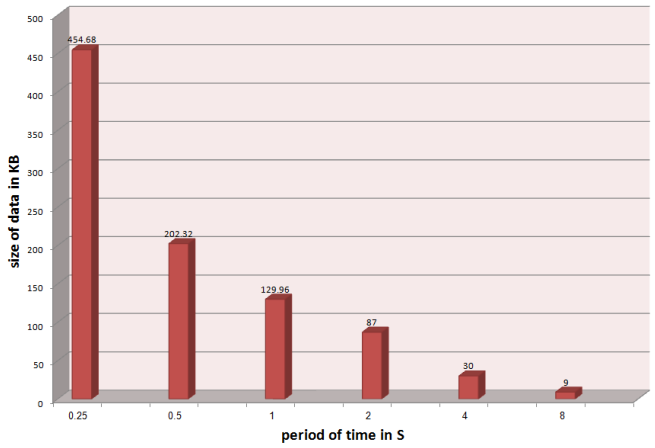

Fig. 5. Effects on Sent data size when using various time periods.

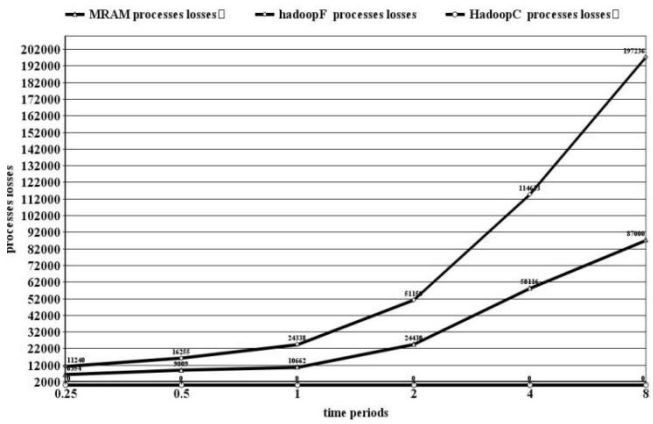

Fig. 6. Processes losses comparison in different platforms.

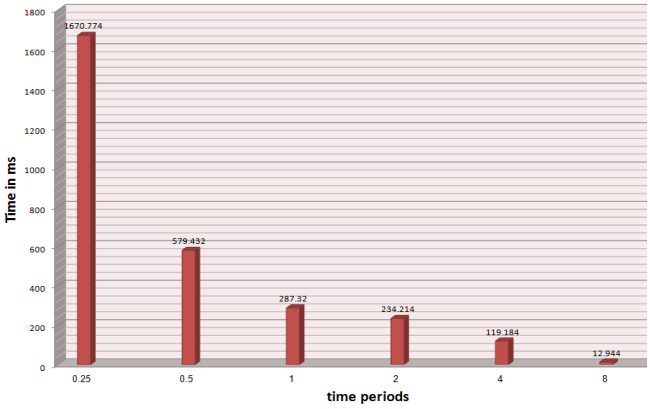

Fig. 7. Effects on Exchange data time when using various time periods.

Based on the above factors, the framework founds the best fixed period of time used in the application is to be transfer data every second. So, the amount of losses processes rates is slightly larger for periods prior to them. In addition, we find the amount of losses processors increases in the periods after which at high rates of up to more than double. When uses a 0.5 second, the losses processes less than a second. But, if we look at other factors we find at 0.5 second the amount of data sent is larger and thus data sent takes a longer period of time to send as shown in Fig.7.

In Hadoop, there are two techniques uses to executing task when a machine is failed. In first technique, the first copy of task starts processing after replicas tasks. But when this machine down the second copy of task is start executing from the beginning. Also, the third copy of tasks is beginning executing when the second machine is failed. The first technique is referred to "HadoopF". In contrast, the idea of second technique is the all duplicates task are start working concurrently after replication process. HDFS takes the first copy of task was executed and cancelled another copies of task. The second technique is referred to "HadoopC".

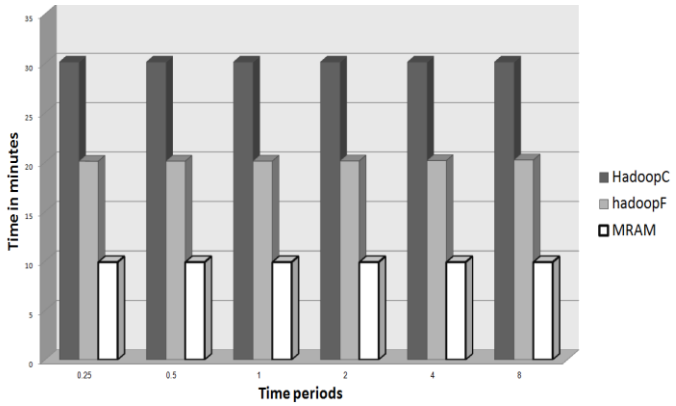

Fig. 8. Cost values for HadoopF, HadoopC and MRAMwhen a machine is failed.

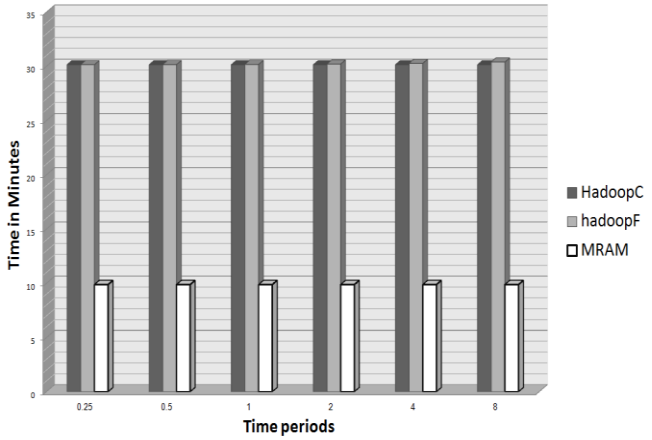

Fig. 9. Cost values for HadoopF, HadoopC and MRAM when two machines are failed

The cost values for MRAM, HadoopF and HadoopC is described in two cases: The first case when one machine is failed shown in Fig. 8 and second case when two machines are failed shown in Fig. 9. The cost is defined by "time". Indeed, the execution time of MRAM is 6.10805 minutes and Hadoop is 6.48875 minutes in optimum system. HadoopF cost is the total time spent in executing task on machines before its failing and this time is referred to (tf). Also, the time spent in executing task on the final machine is added to cost and this time is referred to (ts). In addition to that, the time between the first machine fall and the task beginning in the second machine is added to cost and this time is referred to (tcom). The cost of HadoopF is described by the equation (1).

$$Cost_{HadoopF} = \sum_{Machine=0}^{2} (t_f + t_s + t_{com}) \quad (1)$$

HadoopC cost is completely different from HadoopF because all replicas tasks in HadoopCare working concurrently. In this scenario, HDFS takes the result from the first task has been executed. The time spent from task executed in fastest machine is referred to (T_{fa}). HDFS cancels all others duplicated task and total cost is the summation of time used in all machines until the fastest machine has been finished to executing task. The cost of HadoopC is described by the equation (2).

$$Cost_{HadoopC} = 3 * T_{fa} \quad (2)$$

The cost of MRAM is dependable on the execution time for task (T_e) and total communication time between machines (T_c) and it is described by the equation (3).

$$Cost_{MRAM} = T_e + \sum_{Machine=0}^{2} T_c \quad (3)$$

In Fig.8, the MRAM framework is the lowest cost when one machine is failed. Also, we see that the HadoopF cost is less than HadoopC because the cost of HadoopC is summation of time from all duplicated task in three machines. In HadoopF, the cost value is the summation of execution time in two machines only.

From Fig.9, the MRAM is the lowest cost from HadoopF and HadoopC when two machines are failed. The reason for that, MRAM does not lose the output data and status because there is another copy of them sent to the master machine. Indeed, the cost of HadoopF is larger than HadoopC due to the HadoopF adds the time spent between machine fall and starts task execution on another machine to total cost.

2) Performance Analysis

The word count application is applying on each platform in this step. It is a simple program given a text file and count repeated time for each word, after that save the output as a list in the form of (<Word>, <Count>). It is possible to process each line of a text file completely independently on the other lines. The data then is combined in a central location and the results are printed out. The idea to evaluate the performance of two platforms is measure total time takes to complete assign job. The complete job is executed with different size of data on both Hadoop and MRAM. The execution time for each state is calculated. The load time and mapping task for Hadoop is larger than load time for MRAM because Hadoop takes time for replication processing. It means that each task is sent to three or more machines for fault tolerance, but in MRAM the task sent to only one machine and the mobility supports fault tolerance without needing for replication task. Also, the two platforms using the same algorithm map reducing to evaluate execution time. The total time in MRAM is less than that of Hadoop as shown in Fig.10. MRAM gives the possibility for the server or control node to execute task as another nodes in platform, but not exist in Hadoop.

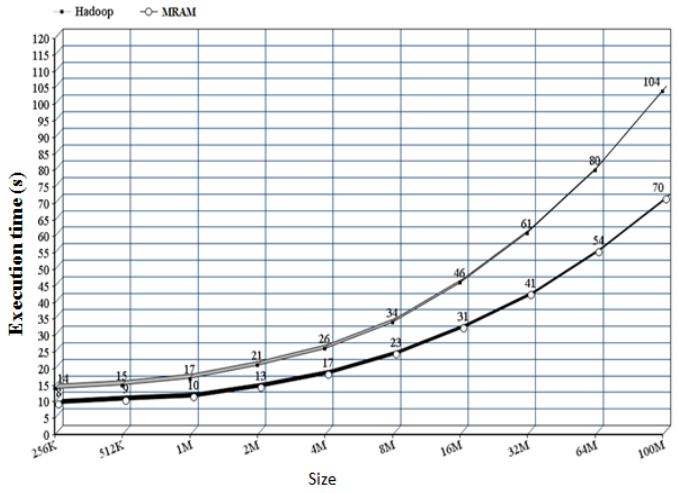

Fig. 10. Performance comparison between MRAM and Hadoop.

3) Centralized Node

This step uses word count application. The total time as shown in Fig.11is composed from the execution time, the time spent from master machine to reconnect again and the time required from new master to work this time is based on the number of times master failed. The value of the time needed from master machine to reconnect is fixed and assumed the master machine is failed one time. This technique is described by the equation (4).

$$T_{Total} = T_{execution} + T_{Migration} + T_m + T_{startup} \quad (4)$$

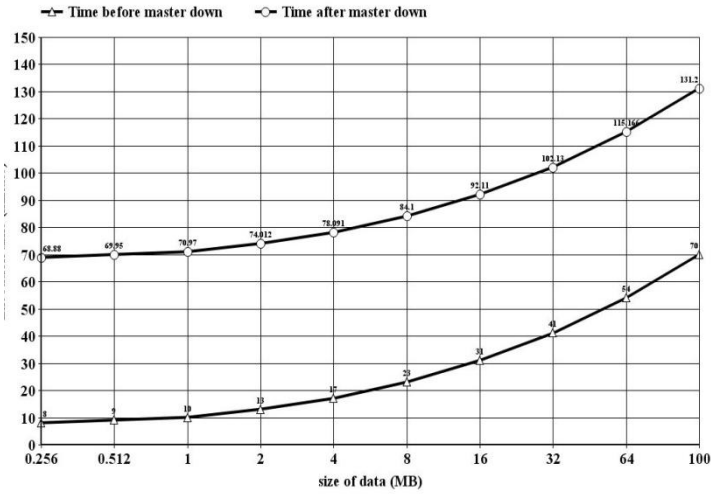

Fig. 11. Performance of MRAM when master machine failed.

Where the Migration time ($T_{Migration}$) is the time required for the agent migrating from the master machine to the target machine and his return. T_m is the time spent from master machine to back online again. The startup time is the time spent from new master to start work, this time is based on the times of master machine failed. The word count application was applied in this step and has been assumed T_m= 1 minute.

4) Summary of results

Table III illustrates the differences between Hadoop and MRAM through the fetched results from experiments. The table shows various comparative factors such as architecture, startup time, performance, reliability, and the mobility support, and … etc.

TABLE III. SUMMARY ABOUT DIFFERENCES BETWEEN TWO PLATFORMS

Factors	Platforms	
	Hadoop	**MRAM**
Architecture	Client/Server	Distributed Agent
Startup time	Long	Less
Performance	Less	Better
Reliability	Reliable	More Reliable
Algorithm	Map-Reduce	Map-Reduce
Mobility	N/A	Support
Management disk	Support	N/A
Allocation Tasks	Support	Support
Scheduling Tasks	N/A	Support
Methodology	Object-Oriented	Object-Oriented
Language	Java	Java

VI. CONCLUSION

In this paper, a new framework called MRAM is developed using mobile agent and Map Reduce paradigm under JADE. Our proposed framework is developed to improve big data analysis and to overcome the drawbacks of Hadoop. In the proposed Framework, mobile agents send both code and data to any machine and react dynamically for any changes in environment. In addition, the mobile agents have ability to move with code and data, if the machine or environment is down. Furthermore, Hadoop is still single master which requires care, this *problem* is solved in MRAM through send met-data contains map of network and all data about tasks and dependences between them. Also, MRAM improves performance by giving the server or control node, the possibility to execute tasks as the others nodes. Another disadvantage of Hadoop, it doesn't support scheduling tasks or does not work with dependent tasks, but MRAM support this feature. A new strategy is written in JAVA programming language based on JADE, This means it can run on different machines and different operating system without any problems.

REFERENCES

[1] Hadoop web site, http://hadoop.apache.org/, Jan. 2014.

[2] Kala Karun. A, Chitharanjan. K, "A Review on Hadoop–HDFS Infrastructure Extensions", In Proceedings of IEEE Conference on Information and Communication Technologies (ICT2013), pp.132-137, 11-12 April, 2013, doi: 10.1109/CICT.2013.6558077.

[3] Jian Tan, Xiaoqiao Meng, Li Zhang, "Coupling Task Progress for MapReduce Resource-Aware Scheduling", In Proceedings of IEEE INFOCOM, pp.1618-1626, 14-19 April, 2013, doi: 10.1109/INFCOM.2013.6566958

[4] Zhu, Nan; Liu, Xue; Liu, Jie; Hua, Yu, "Towards a cost-efficient MapReduce: Mitigating power peaks for Hadoop clusters," Tsinghua Science and Technology, vol.19, no.1, pp.24,32, Feb. 2014 doi: 10.1109/TST.2014.6733205.

[5] Anchalia, P.P.; Koundinya, A.K.; Srinath, N.K., "MapReduce Design of K-Means Clustering Algorithm," International Conference onInformation Science and Applications (ICISA), pp.1,5, 24-26 June 2013, doi:10.1109/ICISA.2013.6579448.

[6] S. Ghemawat, H. Gobioff, and S. Leung. "The google file system", In Proceedings of the nineteenth ACM symposium on Operating systems principles", SOSP '03, pp. 29–43, New York, NY, USA, 2003.

[7] Yongwei Wu; Feng Ye; Kang Chen; Weimin Zheng, "Modeling of Distributed File Systems for Practical Performance Analysis," IEEE Transactions onParallel and Distributed Systems, vol.25, no.1, pp.156-166, Jan.2014, doi:10.1109/TPDS.2013.19.

[8] K.Shvachko, H.Kuang, S.Radia, R.Chansler, "The Hadoop Distributed File System", 26th IEEE symposium on Mass Storage Systems and Technologies (MSST), pp.1-10, 3-7 May, 2010.

[9] JADE web site, http://JADE.tilab.com, Jan. 2014.

[10] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, "JADE: A white paper", TILAB Journal, Vol.3, No.3, pp.6-19, 2003.

[11] D.Schoder, T. Eymann, "Technical opinion: The real challenges of mobile agents", Communications of the ACM, Vol. 43, No. 6, pp. 111-112, June 2000.

[12] G.Mackey, S.Sehrish, J.Wang, "Improving Metadata Management for Small Files in HDFS", In Proceedings of IEEE International Conference on Cluster Computing and Workshops, pp.1-4, 31 Aug.- 4 Sept., 2009.

[13] J. Dean, S. Ghemawat, "MapReduce: A Flexble Data Processing Tool", Communications of the ACM, Vol.53, No.1, pp.72-77, January, 2010.

[14] V.Martha, W.Zhao, Xiaowei Xu, "h-MapReduce: A Framework for Workload Balancing in MapReduce", IEEE 27th International Conference on Advanced Information Networking and Applications, pp.637-644, 25-28 March, 2013.

[15] J.Shafer, S.Rixner, Alan, "The Hadoop Distributed Filesystem:Balancing Portability and Performance", In Proceedings of IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS), pp.122–133, 28-30 March, 2010.

[16] K. Shvachko, H. Kuang, S. Radia, R. Chansler, "The Hadoop Distributed File System", 26th IEEE Symposium on Mass Storage Systems and technologies(MSST), pp. 1-10, 3-7 May, 2010.

[17] C.Lam, "Hadoop in Action", Manning Publications Co., USA,ISBN:9781935182191, Dec. 2010.

[18] S.Perera, T.Gunarathne, "Hadoop MapReduce Cookbook", Packt Publishing, ISBN:1849517282,Jan. 2013.

[19] D. Lange, M. Oshima, "Programming and Deploying Java Mobile Agent with Aglets", Addison-Wesley, pp.18-20, 1998.

[20] P. Braun and W. Rossak, "Mobile Agents – Basic Concepts Mobility Models and the Tracy Toolkit", Morgan Kaufmann Publishers, 2005.

[21] L. Guanyu ; W. Baofeng; Y. Yang; A. Lihua , "Researches on Performance Optimization of Distributed Integrated System Based on Mobile Agent", The Sixth World Congress on Intelligent Control and Automation, pp. 4038- 4041, June 2006.